Please replace the paragraph beginning at page 1, line 15, with the following rewritten paragraph:

-- The graphics pipeline maps, or renders, each primitive into a memory storage device known as a frame buffer. Each storage location within the frame buffer defines one pixel within the image being produced. The graphics pipeline performs the rendering process by determining which pixels (i.e., which frame buffer storage locations) are included within each primitive. Each pixel is then initialized to reflect the attributes of the primitive, or primitives in which it is included. In many cases, the graphics pipeline will further modify the pixel values in the frame buffer to apply texture, lighting and other effects to the graphics primitives. --

Please replace the paragraph beginning at page 3, line 7, with the following rewritten paragraph:

-- The present invention provides a method and apparatus for early occlusion culling. For the present invention, the screen is divided into a series of tiles arranged as a rectangular grid. The rectangular grid is known as a coarse Z-buffer and may have various sizes and dimensions. For the purposes of this description, a size of two-hundred and fifty-six tiles arranged in a sixteen by sixteen grid may be assumed. Each tile within the coarse Z-buffer has an associated depth value. Each tile's depth value is defined as the farthest Z-buffer value that is included within that tile. --

Please replace the paragraph beginning at page 4, line 3, with the following rewritten paragraph:

-- The depth values are stored in a location, such as main memory, where they are available to application programs. This allows application programs to reference these values while they are creating graphics images. The program rendering an image constructs a surrogate volume for each object that it adds to the image. The program then compares the nearest Z-value of the surrogate volume to the depth value of the tile that includes the surrogate volume. Based on this comparison, the application program determines if the object is occluded and can be discarded. --

Please replace the paragraph beginning at page 5, line 5, with the following rewritten paragraph:

a5 cont.

-- Advantages of the invention will be set forth, in part, in the description that follows and, in part, will be understood by those skilled in the art from the description herein. The advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims and their equivalents. --

Please replace the paragraph beginning at page 7, line 7, with the following rewritten paragraph:

a6 cont.

-- In Figure 1, a computer system 100 is shown as a representative environment for the present invention. Structurally, computer system 100 includes a host processor 102 and a memory 104. An input device 106 and an output device 108 are connected to host processor 102 and memory 104. Input device 106 and output device 108 are connected to host processor 102 and memory 104. Input device 106 and output device 108 represent a wide range of varying I/O devices such as disk drives, keyboards, modems, network adapters, printers and displays. Each node 102 may also include a disk drive 110 of any suitable type (equivalently, disk drive 110 may be any non-volatile mass storage system such as "flash" memory). Computer system 100 also preferably includes a graphics processor 112 of any suitable type. Graphics processor 112 implements all of the tasks required to translate graphics primitives and attributes to displayable output. In Figure 1, host processor 102 and graphics processor 112 are interconnected using a bus. It should be appreciated that the present invention is equally suitable to environments where host processor 102 and graphics processor 112 share a commonly addressable memory. --

Please replace the paragraph beginning at page 8, line 1, with the following rewritten paragraph:

a7 cont.

-- Computer system 100 is the hose for a graphics pipeline. An implementation for this pipeline is designated 200 in Figure 2. Pipeline 200 includes generation stage 202, traversal stage 204, transformation stage 206, rasterization stage 208 and display stage 210. Generation stage 202 corresponds to the creation, acquisition, or modification of information to be displayed and organizing this information into application data structures. Traversal stage 204 corresponds to the traversal of the application data structures generated in the preceding stage, passing on the appropriate graphics data. Transformation stage 206 corresponds to the transformation of the graphics data from object-space coordinates into eye-space coordinates, performing requested

lighting operations, clipping the transformed data in clip-space- , and projecting the resulting coordinates into window space. Rasterization stage 208 renders window space primitives (such as points, lines, and polygons) into a frame buffer. Per-vertex shading calculations, texture lookups and calculations, and per-pixel operations such as depth testing are performed in this stage. Display stage 210 scans the resulting pixels in the frame buffer, typically for display to a video monitor or other device. --

Please replace the paragraph beginning at page 9, line 7, with the following rewritten paragraph:

--The present invention provides a method and apparatus for early occlusion culling. The method and apparatus of the present invention are better understood by reference to representative image 300 of Figure 3. Image 300 depicts a runway partially obscured by clouds. Eye-point 302 represents the position at which image 300 is viewed. Eye-point 302 is intended to be movable in relation to image 300. This means that components of image 300, including the runway and clouds may be viewed from a range of positions. --

Please replace the paragraph beginning at page 10, line 3, with the following rewritten paragraph:

-- Graphics pipeline 200 is configured to update the depth values in memory 104. This means that, for the particular configuration shown, rasterization stage 208 uses feedback loop 212 to continuously update the depth values within memory 104. Different configurations may perform this update as either a "push" or "pull" operation. Thus, for some configurations, rasterization stage 208 will transfer (push) depth values to memory 104. In other configurations, another entity (such as traversal stage 204 or the graphics application program executing on host processor 102) will retrieve (pull) depth values from rasterization stage 208. The depth values may also be updated either synchronously or asynchronously. For synchronous updating, depth values are transferred to memory 104 as they change within rasterization stage 208. Typically this means that depth values are updated each time corresponding Z-values in a Z-buffer used by rasterization stage 208 are changed. Graphics pipeline 200 may also be configured to perform these updates on a less frequent, asynchronous basis. In many cases this means that graphics pipeline 200 will perform these updates on a periodic basis. In other cases, graphics pipeline 200